VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Částicový systém implementovaný pomocí technologie CUDA

CUDA Powered Particle System

2013 Karel Konupčík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Karel Konupčík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Částicový systém implementovaný pomocí technologie CUDA
CUDA Powered Particle System

Zásady pro vypracování:

Cílem bakalářské práce je využít technologii NVIDIA CUDA pro výpočet simulace částicového systému. Student systém implementuje za použití technik masivní paralelizace a simulaci graficky znázorní. Závěrem práce bude diskuze o výkonu implementace a přínosu masivní paralelizace oproti standardním programovacím technikám.

Bakalářská práce musí splňovat následující body:
1. Úvod do problematiky částicových systémů.
2. Krátký přehled metod řešení problému.
3. Implementace zvoleného částicového systému.
4. Vizualizace částic a jejich interakcí.
5. Analýza a vyhodnocení výkonnosti implementace.

Seznam doporučené odborné literatury:

[1] Allen Sherrod. Game Graphics Programming. Charles River Media, 2008
Dále dle pokynů vedoucího bakalářské práce

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Dohnálek**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě, dne 7. května 2013 Karel Konupčík .....

Karel Konupčík

Poděkování

Rád bych touto cestou vyjádřil poděkování Ing. Pavlovi Dohnálkovi za jeho cenné rady a trpělivost při vedení mé bakalářské práce.

Abstrakt

Bakalářská práce se zabývá problematikou částicových systémů včetně jejich využití. Částicové systémy slouží k modelování "fuzzy objektů", které se skládají z mnoha malých částic. Uvedené objekty by těžko byly tvořeny jiným způsobem, protože nemají pevný povrch. Pomocí částicových systémů lze vytvořit zajímavé efekty, jako jsou výbuchy, kouř, plameny. Tyto efekty mají široké využití od filmového průmyslu, až po herní průmysl. V bakalářské práci je implementován částicový systém pomocí technologie CUDA, kde je simulován pohybový efekt barevných částic. V implementaci je využito hlavní výhody CUDA technologie, tedy masivního paralelismu, což poskytuje velmi dobrý výkon i pro velké množství částic.

Klíčová slova

Částicové systémy, Částice, CUDA, GPU programování, OpenGL, Vizualizace částic, Kernel

Abstract

This bachelor thesis is focused on the particle system problem and its practical use. Particle systems are used to create "fuzzy object" which are objects created from many small particles. It would be difficult to create them by any other means because fuzzy object have no solid surface. Particle systems can be used to create interesting effects like explosions, smoke or flames. These effects have a variety of use, from movie to game creation. In this bachelor thesis, a particle system is implemented with the CUDA technology where a movement effect of coloured particles is simulated. The implementation takes advantage of the CUDA technology's main advantage - massive parallelism. This provides very high performance even for large amounts of particles.

Key Words

Particle systems, Particles, CUDA, GPU programming, OpenGL, Particle visualisation, Kernel

Seznam použitých zkratek

API Aplication Programming interface
CPU Central processing unit
CUDA Compute Device Unified Architecture
FPS Frames Per Second
PC Personal Computer
GPU Graphic ProcessingUnit
GPGPU General Purpose Computing on Graphic Processing Unit
OpenGL Open Graphic library
RGB Red Green Blue
Sci-fi Science Fiction
VBO Vertex Buffer Object

Seznam použitých symbolů

AAA Označení pro PC herní tituly vysoké kvality s velkým rozpočtem

Obsah

1 Úvod do problematiky částicových systémů

Pojem částicový systém je možné v oblasti počítačové grafiky definovat jako soubor modelovacích, renderovacích a animovacích technik, jejichž výsledkem je zobrazení určitých efektů, které nemají pevný povrch, většinou nejsou statické a v průběhu času se mění a vyvíjí.

1.1 Částicový systém obecně

Částicový systém je tvořen velkým počtem malých objektů, částic, které jsou reprezentovány jako bod v 3D prostoru. Interakce částic s okolím a mezi sebou je dána předem danými pravidly tak, aby vytvářely požadovaný efekt nebo objekt. Typický částicový systém bývá většinou reprezentován v 3D prostředí. 2D částicové systémy nejsou příliš časté.

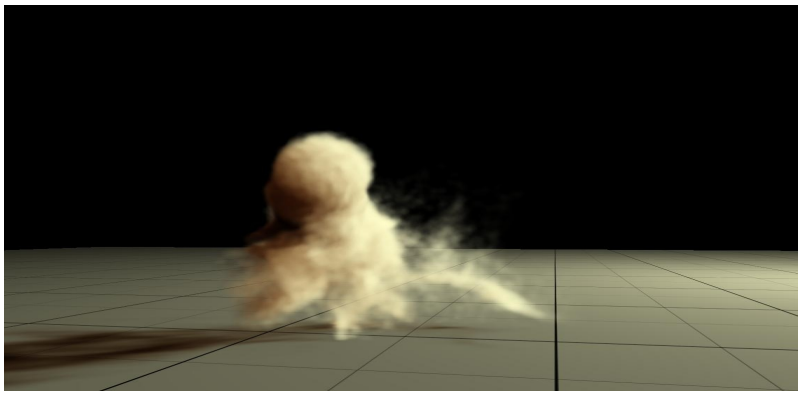
1.2 Využití částicových systémů

Částicové systémy jsou pro svoji flexibilitu a schopnost zobrazit jakýkoliv efekt nebo objekt, složený z jednotlivých bodů, hojně využívány. Mohou být použity k vytvoření speciálních efektů ve filmových dílech, k simulaci různých fyzikálních jevů, nebo v herním průmyslu pro oživení her realistickými efekty, jako výbuchy, déšť, sněžení a podobně.

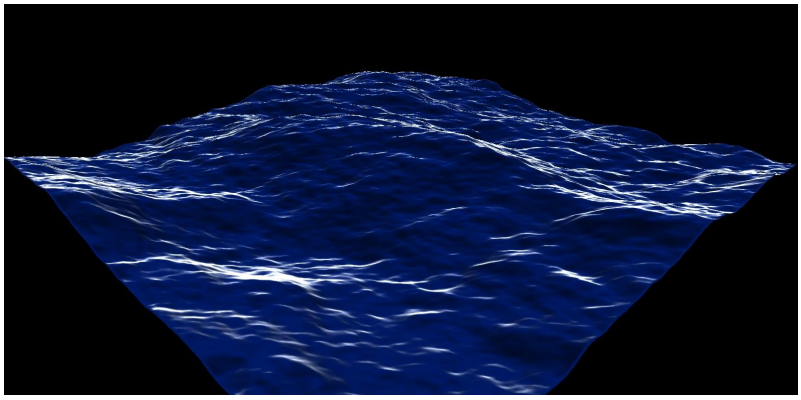
1.2.1 Částicové systémy pro simulaci

Částicové systémy mohou být využívány pro názornou simulaci různých fenomén, kdy je vytvořena scéna, ve které se pohybuje velké množství částic přesně danými pravidly simulující požadovaný efekt. Mohou to být efekty sypkých materiálů, pohybu tekutin a vůbec čehokoliv, co lze poskládat z jednotlivých částic.

Uvedené efekty slouží ke znázornění pohybových jevů bez nutnosti vyvolání skutečné reálné akce. Pomocí technologie částicových systémů lze zobrazit i jevy ve skutečnosti neproveditelné z technického, časového, finančního i jiného jakéhokoliv důvodu. Praktické využití lze nalézt v mnoha oblastech současného života. V průmyslu může jít o vývoj nových výrobních technologií, s ukázkou dopadu na konečný produkt. V lékařství je možné využít zobrazení činností související s pohybem i samotnou existencí člověka z pohledu stavby těla k určení optimálního procesu při léčení pacientů. Simulace se stala prostředníkem při výzkumu nejrůznějších jevů v mnoha oblastech života. Je podporou pro výzkumy fyzikálních, chemických a jiných stávajících procesů. Slouží k vysvětlení jinak nerealizovatelných jevů na zemi i ve vesmíru a vědě pomohly otevřít nové možnosti k přesnějším závěrům při výzkumných činnostech. Přínosem je využívání simulací ve školství a při vzdělávání obyvatelstva.



Ilustrace 1: Zobrazení simulace kouře, CUDA examples: smokeParticles



Ilustrace 2: Zobrazení simulace vodní plochy, CUDA examples: oceanFFT

1.2.2 Částicové systémy ve filmovém průmyslu

Částicové systémy jsou velmi často využívány také filmovým průmyslem ve formě speciálních efektů. Používají se například pro zobrazení explozí, jejichž provedení by bylo ve skutečnosti za pomoci demoličních expertů na výbušniny velmi nákladné nebo naprosto nemožné. Tvůrci s podporou filmových studií za pomoci počítačové animace vytváří požadované efekty, s následným vložením do filmového díla. Výbuchy jsou jen zlomkem ve výčtu simulovaných procesů, jaké specialisté přes animaci v rámci částicových systémů pro film vytváří. Je mnoho typů akcí, které jsou vytvořeny výhradně prostřednictvím animace. Může být simulována střelba z různých sci-fi zbraní a podobně. Přitom posuzování kvality efektu je velmi

relativní. Zobrazení skutečných efektů je nepochybně náročnější, než simulace fantazijních jevů a pohybů. Například zobrazení reálných plamenů a výbuchů je obtížnější, než animování střelby z vesmírné lodi obklopené energetickým štítem. Je to dáno skutečností, že diváci již určitě ve skutečnosti viděli, alespoň v televizi nebo kdekoliv jinde, jak skutečný neanimovaný výbuch nebo plameny vypadají a mají možnost porovnání daných jevů. Nikdy však nespatřili vesmírnou loď střílet z neexistující energetické zbraně a možnost porovnání zde chybí. Z toho plyne, že při stejné úrovni kvality efektu bude střílející vesmírná loď skoro vždy z pohledu diváka vypadat lépe, než animovaný výbuch čehokoliv reálného.
Jako ukázka filmových sci-fi efektů slouží ilustrace 3:

Ilustrace 3: Simulované výbuchy ze seriálu Battlestar Galactica

1.2.3 Částicové systémy v herním průmyslu

Částicové systémy pro simulaci a ve filmovém průmyslu mohou být poměrně hodně využívané, u počítačových her se staly pomalu standardem. Při simulaci částicový systém simuluje určitý proces, u kterého se zadaný akt poměří, zaznamená, vypočítá, a tím splní svůj úkol. Ve filmové tvorbě se vytvoří scéna s částicovým systémem a po narenderování se vloží do filmu. Částicové systémy ve hrách bývají často komplexnější, mohou na sebe vzájemně reagovat, disponují interakcemi mezi prostředím a hráčem. Moderní AAA herní tituly by se v dnešní době bez určité míry využití částicových systémů neobešly. Výsledná hra bez použití jakýchkoliv forem částicových systémů by vypadala podivně, nedodělaně, bylo by patrné, že jí něco schází. Hry s absencí částicových systémů si mohou tvůrci dovolit u nezávislých her, které vytváří několik jedinců nebo malá skupina lidí a pracují s poměrně malým rozpočtem. Vydání AAA titulu bez jakýchkoliv částicových systémů je však nereálné. I kdyby se tvůrci pokusili hru pozvednout jinými aspekty, výsledný produkt by nemusel naplnit představy hráčů očekávajících

slušné grafické zpracování, k čemu je použití částicových systému téměř nezbytné. Zde jsou příklady zobrazující výše uvedené částicové systémy v dnešních počítačových hrách.

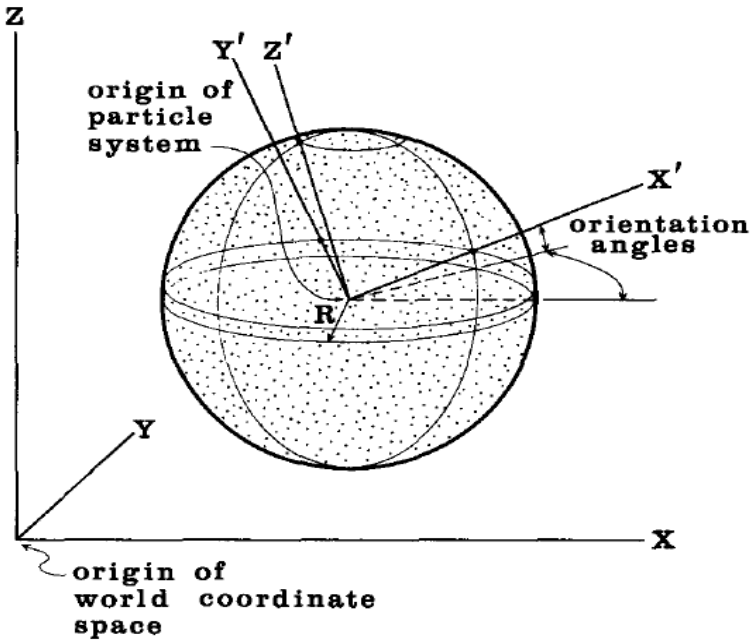
Ilustrace 4: Částicový systém kouře ve hře Planet Side 2

Ilustrace 5: Částicový systém sněžení ve hře Skyrim

1.3 Model částicového systému

Částicový systém se skládá z mnoha částic, které společně vytváří určitý objekt. Průběžně jsou částice vygenerovány do systému, v systému se pohybují, a také z něj mizí. Každý krok v průběhu života částicového systému vykonává stanovené akce. Jsou vytvořeny

nové částice, a ty jsou přidány do systému, kde každá částice má nějaké atributy. Všechny částice, kterým z nějakého důvodu skončila životnost, (v důsledku dosažení maximálního věku, nebo například přesáhly okraj vykreslované scény) jsou z částicového systému odstraněny. Ostatní částice obsažené v systému pokračují v existenci a jejich atributy jsou měněny podle toho co má konkrétní částicový systém za úkol. Posledním krokem je vykreslení celé scény, kdy je výsledný obraz vytvořeného efektu prezentován uživateli.



Ilustrace 6: Model částicového systému [1]

1.4 Částice

Částice je základním stavebním kamenem každého částicového systému. Každá částice uvnitř částicového systému prochází životním cyklem. Nejdříve je vytvořena a umístěna do systému, poté proběhne její život, kdy jsou měněny některé nebo všechny její atributy. Na závěr částice zanikne, ať už proto, že ji vypršel čas, po který měla existovat, nebo z jiného důvodu, a je z částicového systému odstraněna. Tím je uvolněno místo pro vznik nové částice.

1.4.1 Atributy částic

Každá částice uvnitř klasického částicového systému má atributy, které definují její vzhled, chování a existenci uvnitř částicového systému. Mezi tyto atributy mohou patřit následující:

Pozice - Pozice slouží pro určení umístění částice uvnitř částicového systému. Je dána souřadnicemi <x, z, y>, čímž je určena pozice v 3D prostoru. Pozice bývá zadána při vytvoření částice a následně bývá upravována za běhu programu.

Rychlost - Je rychlost, jakou se částice pohybuje. Rychlost může být dána při vytvoření částice, ale také nemusí. Rychlost se mění za běhu programu, může být konstantní, nebo může být postupně upravována.

Životnost/věk - Tento atribut ovlivňuje, po jakou dobu bude částice aktivní uvnitř částicového systému. V případě věku se atributu přičítá nějaká určená hodnota. Jakmile částice dosáhne maximálního věku je odstraněna a uvolní místo pro vytvoření částice nové. V případě životnosti je částici dána určitá hodnota a ta je zmenšována. Jakmile dosáhne hodnoty 0, je důsledek stejný, jako v případě, kdy věk dosáhne maximálního věku.

Aktivní - Atribut aktivní je vlastnost, která určuje, jestli je částice aktivní, jestli její životnost neklesla pod nulu nebo nepřesáhla zadaný maximální věk. Nabývá dvou hodnot, jestli je částice aktivní nebo není.

Hmotnost - Váha částice, bývá použita při vypočítávání změny rychlosti vlivem gravitační síly.

Textura - Na plochu částice je načtena textura, která ji pokryje. Využití textur je nutnost pro tvorbu realistických efektů.

Barva - Atribut barva určuje zbarvení částice. Barva je tvořena pomocí barvového modelu RGB. Tyto tři základní barvy mohou svými kombinacemi vytvořit všechny ostatní barvy. Jednotlivé složky RGB mohou nabývat hodnot od 0 až po 255.

Velikost - Fyzická velikost částice při zobrazení.

Průhlednost - Průhlednost částice, v propojení s věkem lze vytvořit zajímavé efekty. Například čím vyšší bude věk u částice, tím průhlednější bude. Až se bude blížit maximálnímu věku, tak bude téměř úplně průhledná.

1.4.2 Životní cyklus částice - vytvoření částice

Nové částice jsou do částicového systému přidány pomocí kontrolovaných náhodných metod. Hlavním parametrem je, kolik celkem částic se objeví na scéně. Množství generovaných částic se odvíjí od požadavku na hustotu cílového efektu. Částice mohou být přidávány dvěma způsoby.

První metodou je, že v každém kroku je přidáno určité fixní číslo částic, nebere se ohled na nic jiného. Předpokládá se, že starším částicím vyprší jejich životnost a ze scény zmizí. Pokud by se tak nestalo, scéna by byla přehlcena částicemi a výsledný efekt by pravděpodobně nevystihoval původní cíl.

Druhou metodou je, že se zjistí, kolik částic je na scéně. Teprve pak je rozhodnuto kolik a jestli vůbec budou nějaké částice přidány. Tato metoda se výhodnější, pokud vzniká potřeba kontrolovat přesný počet částic, které chceme zobrazovat.

1.4.3 Životní cyklus částice - vývoj částice

Jednotlivé částice uvnitř částicového systému se pohybují v 3D prostoru, mění svoji rychlost, barvu, průhlednost, velikost a podobně. Záleží na jednotlivých atributech a požadavcích na částicový systém jako celek. Mezi jednotlivými snímky se obvykle k poloze částice přičítat nějaká hodnota, o tolik bude měnit svou pozici v následujícím snímku. Tím je vlastně zajištěn samotný pohyb částice. Pro přidání komplexnosti může být přidána i akcelerace nebo decelerace, kdy se částice pohybuje rychleji nebo pomaleji s každým dalším snímkem. Tahle vlastnost může být použita například pro implementaci gravitace, která budou působit na částice.

1.4.4 Životní cyklus částice - zánik částice

Každá částice uvnitř částicového systému má vlastní určitou životnost. Jakmile částicový systém běží, je tato životnost snižována nejlépe o náhodnou hodnotu v rozumném rozmezí tak, aby částice všechny nezanikaly a nevznikaly v ten samý moment. Jakmile životnost částice dosáhne nuly, je z částicového systému odstraněna. Vypršení životnosti ale nemusí být jediný důvod odstranění částice ze systému. Dalším důvodem může být například přesun částice za vykreslovanou scénu, kde je vlastně zbytečná, a je proto odstraněna. Výše uvedené situace jsou obecně nejpoužívanějšími důvody pro odstraňování částic z částicového systému. Částice mohou být odstraněny i z jakékoliv jiné příčiny, žáleží na implementaci konkrétního částicového systému, faktorem může být klidně i barva [1, 2].

1.5 Fáze částicového systému

Částicový systém jako celek prochází dvěma fázemi. V první fázi se systém zaplní jednotlivými částicemi, provedou se veškeré potřebné výpočty a změny atributů částic. Ve druhé fázi je výsledek zobrazený uživateli.

1.5.1 Simulační fáze

Během simulační fáze je vytvořeno určité množství částic, u kterých jsou provedeny všechny potřebné výpočty, jako jak rychle se mají vytvářet, na jaké pozici, jsou vypočítány pohyby částic. Každá částice je poté umístěna podle souřadnic do 3D prostoru dle startovní pozice. Jsou také zpracovány veškeré atributy částice, jako je například barva, rychlost, životnost atd. Veškeré existující částice jsou zkontrolovány, jestli nepřekročily daný věk nebo jim nedošla životnost. Pokud tato situace nastane, jsou všechny částice s prošlou životností nebo překročeným věkem z částicového systému odstraněny a uvolní místo novým částicím. Protože částicové systémy většinou pracují s předem pevně daným počtem částic, tak odstranění starých částic je nutné z důvodu vytvoření prostoru pro vznik částic nových. Pokud životnost částice nevypršela, dostává se na řadu určitý specifický složitý nebo jednoduchý matematický výpočet, který posune pozici částice tak, aby částicový systém vytvořil požadovaný efekt. Pohyb může být realizován pomocí jednoduchého přičítání fixní hodnoty k aktuální pozici částice, postupným zvyšováním rychlosti částice až k výslednému stupni rychlosti, nebo matematickými operacemi simulujícími reálně fyzikální jevy v podobě gravitace, externí síly působící na částice ve všech směrech, interakce mezi částicemi atd. Běžně používanou praktikou k tvorbě částicového systému může také být používání kolizí. Kolize v částicovém systému zajišťují částic od různých objektů na scéně, samy od sebe navzájem, nebo od okrajů vykreslované scény. Záleží na implementaci, v jaké míře budou kolize využity. Kolize jsou výpočetně poměrně náročné v porovnání s ostatními operacemi v částicovém systému. Při jejich použití je třeba s tímto faktem počítat a používat je jen v opravdu nutných případech, nebo jen v situacích, kde příliš nezáleží na výpočetní náročnosti výsledného částicového systému. Částicový systém také může také mít sadu pravidel, které jsou aplikovány na všechny částice. Tato pravidla mohou být propojena s životností jednotlivých částic. Například částicový systém může být implementován s cílem snižování průhlednosti jednotlivých částic spolu s jejich životností. Průhlednost se bude u konkrétních částic snižovat až do jejich úplného zmizení [3].

1.5.2 Renderovací fáze

Jakmile jsou provedeny všechny potřebné matematické výpočty, jsou výsledná data poslána jako update na renderování. Každá částice bude vyrenderována jako samostatná entita. Pojem renderování znamená vytváření reálného obrazu na základě počítačového modelu, jedná

se tedy o vlastní vykreslení scény. Částice bývá obvykle vyrenderována ve formě 3D objektu. Nicméně nejde o pravidlo. Částice může být klidně zobrazena i jako pixel ve 2D prostředí. Výše uvedený způsob je vhodný v těch případech, kdy je k dispozici limitovaná výpočetní kapacita [3].

2 Přehled metod řešení problému

2.1 GPU programování

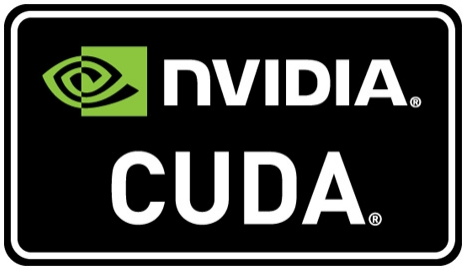
GPU programovaní jinak nazývané GPGPU je využívání GPU grafické karty pro výpočty, které jsou jinak prováděny pomocí CPU. GPU tedy i přes to že je primárně určeno na grafické výpočty, může pomocí této techniky plnit stejnou výpočetní funkci jako CPU [4].

2.1.1 GPU programování - historie

První GPU byly navrženy jako grafické akcelerátory, které plnily přesně předem danou úlohu. V devadesátých letech hardware začal být více programovatelný, což vedlo k první grafické kartě obsahující GPU. Jednalo se o grafickou kartu GeForce 256, která byla vydána 11.října 1999. Vydání této karty otevřelo dveře nové éře grafiky. GPU bylo vítanou novinkou, kterou přivítali nejen herní vývojáři pro implementaci grafiky nové generace do svých počítačových her. Začala být využívána také pro svůj výpočetní výkon – GPGPU. GPGPU bylo ve svých počátcích extrémně složité, jen nepočetná hrstka lidí byla schopná jejího využití. To se změnilo v roce 2003, když skupina výzkumníků na Standforské univerzitě pod vedením vědce jménem Ian Buck Brook umožnila použití jazyka C s paralelním využitím dat. S využitím streamů, kernelů a redukčních operátorů začalo opravdové používání GPU jako regulérního procesoru CPU. To vedlo k tomu, že tvorba programů byla méně komplikovaná, i jejich ladění bylo o mnoho snadnější. Také výkon takto vytvořené aplikace byl přibližně sedmkrát vyšší. Firma NVIDIA Brooka najala za účelem vytvoření intuitivní metody na programování CPU. Výsledkem bylo že v roce 2006 NVIDIA vydala model CUDA, první GPU programovací model pro širokou veřejnost [5].

2.2 CUDA

CUDA je paralelní výpočetní platforma a programovací model rozšiřující jazyk C, vyvinutý a vydaný společností NVIDIA.



Ilustrace 7: Logo CUDA, NVIDIA

2.2.1 CUDA - úvod

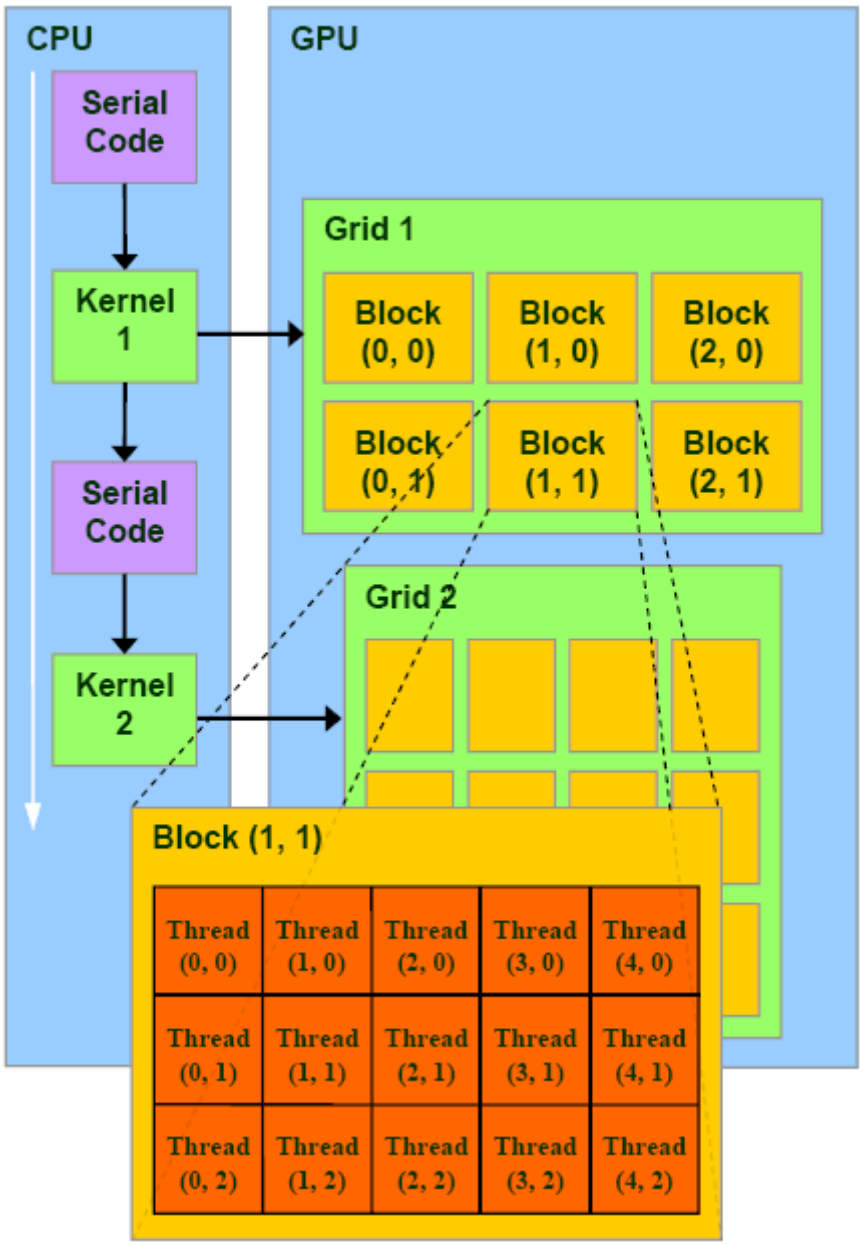
Technologie CUDA umožňuje využívání výpočetního výkonu grafické karty (GPU). Představuje soubor nástrojů, který umožňuje programátorům spouštění kódu v jazyku C za využití GPU grafické karty místo klasického CPU. Hlavní výhodou využití GPU na rozdíl od CPU je, že grafické karty jsou cíleně vyráběny k zvládání masivního paralelismu. Jedná se o vlastnost výborně využitelnou pro řešení určitých typů problémů. Typické využití je například při potřebě vykonávat zadaný úkol opakovaně, s různými daty pro každý průběh. Klasické CPU zvládne zpracování jen nepočetného množství vláken najednou. Na druhou stranu technologie CUDA za použití grafické karty GPU zvládá až tisíce vláken spuštěných zároveň [5, 6].

2.2.2 CUDA – model

Základní myšlenkou technologie CUDA je využití velkého množství paralelně běžících vláken – masového paralelismu v místech, za pomoci zpracování dat na GPU grafické karty. Zpracovaná data jsou za běhu programu dynamicky dostupná k dalšímu využití. Mezi klasické využití masivního paralelismu patří zpracování obrazu, simulování fyzikálních modelů jako může být simulace kapalin, simulace finančních modelů a analýz, třídění, vyhledávání. Nicméně úkoly, které vyžadují složitější datové struktury jako stromové struktury nebo asociativní pole, jsou na GPU vykonávány pomaleji nežli na CPU. Současné GPU totiž s daty pracují pomocí klasických polí. Proto pro vytvoření efektivní implementace je GPU zpracování využíváno jen na jednotlivé úkoly zaměřené na zpracování velkého množství dat. Části kódu běžící na GPU se nazývají kernely. Kernel tedy není aplikací, je ale hlavním výpočetním prvkem v každém obsáhlém kroku zpracování aplikace. Typická aplikace s použitím GPU obsahuje kernely a klasický sériový kód [11]. Sériový kód bývá umístěn mezi jednotlivými kernely. Kernel je prováděn každým spuštěným vláknem. Organizace vláken na GPU vypadá takto:

Blok – vlákna jsou organizována do jednorozměrných (1D), dvourozměrných (2D) nebo trojrozměrných (3D) bloků. Vlákna v každém jednotlivém bloku mohou sdílet data a také je možné synchronizovat jejich běh. Každý blok vláken musí být schopen pracovat nezávisle na ostatních blocích, aby byla umožněna rozšiřitelnost sytému (GPU obsahující více jader dokáže spustit více bloků než GPU s menším množstvím jader). Počet vláken v jednotlivém bloku je určen technickými možnostmi zařízení. Každé vlákno v rámci jednoho bloku je ve spuštěném kernelu přístupné přes zabudovanou proměnnou threadIdx.

Mříž – jednotlivé bloky jsou organizovány do jednorozměrné (1D), dvourozměrné (2D) nebo trojrozměrné (3D) mříže. Bloky je možno v rámci mříže ve spuštěném kernelu identifikovat přes zabudovanou proměnnou blockIdx [7].



Ilustrace 8: Průběh vykonávání programu, kernel, mříže a bloky
http://www.realworldtech.com

2.2.3 CUDA - paměťový model

Technologie CUDA má kromě vykonávacího modelu také svůj vlastní paměťový model, který má přístup do různých druhů pamětí. Každá paměť má svoje výhody a omezení, na které se musí brát ohled při vytváření CUDA kernelu. Každé zařízení schopné pracovat s CUDA poskytuje několik typů pamětí s rozdílnými parametry, jako je například latence nebo

adresovací prostor. Konkrétně se jedná o paměti: registr, sdílená paměť, lokální paměť, globální paměť, paměť konstant a paměť textur.

Registrová paměť - přístup do této paměti je velmi rychlý, maximální počet registrů na každý blok je omezen. Proměnné v registru jsou privátní k jednotlivým vláknům. Každé vlákno v jednom bloku dostane privátní verzi každé proměnné z registru. Proměnné v registru existují jen tak dlouho, dokud je vlákno aktivní. Jakmile je vlákno ukončeno, tak k proměnné už nebude moci být znovu přistoupeno.

Lokální paměť - paměť je využívána, jakmile dojde k vyčerpání paměti registrů. Stejně jako u registrové paměti je i lokální paměť privátní k vláknu. Každé vlákno musí inicializovat obsah proměnných předtím, než mohou být použity. Proměnné v lokální paměti mají stejnou životnost jako vlákno, ve kterém jsou využívány. Jakmile vlákno skončí, je obsah proměnné nedostupný.

Sdílená paměť - přístup do sdílené paměti je velmi rychlý, přibližně 100krát rychlejší než u paměti globální. Nicméně její kapacita je nízká. Mohou do ní přistupovat všechna vlákna v daném bloku. Proměnné ve sdílené paměti mají životnost jako blok, na rozdíl od registrové a lokální paměti, které měly životnost jako vlákno. Změny provedené v této paměti musí být synchronizovány pomocí __syncthreads() uvnitř kernel funkce, aby byla zajištěna integrita dat.

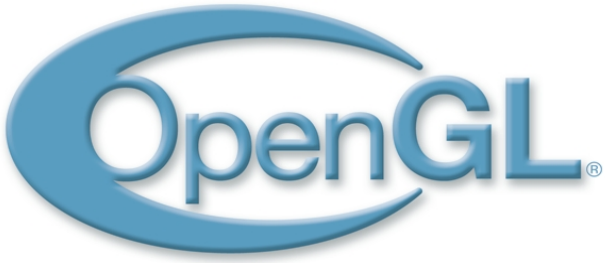
Globální paměť - tato paměť je pomalejší, disponuje však největší kapacitou. Její životnost je stejná jako životnost celé aplikace a může být přistupována všemi vlákny v kernelu. Je deklarována na hostu pomocí příkazu CUDAMalloc a uvolněna příkazem CUDA Free. Do globální paměti může být zapisováno a čteno z hosta pomocí příkazu CUDAMemcpy.

Paměť konstant - je použita pro konstanty, jako jsou globální proměnné, konstantní hodnoty a jiné. Paměť je přístupná z hosta za použití příkazů CUDAMemcpyToSymbol pro čtení a CUDAMemcpyFromSymbol pro zápis [8].

Paměť	Umístění	Vyrovnávací paměť(Cache)	Přístup:Host	Přístup:kernel	Působnost	Životnost
Registrová	Na čipu	ne	ne	Čtení/Zápis	Vlákno	Vlákno
Lokální	Mimo čip	2.x - ano	ne	Čtení/Zápis	Vlákno	Vlákno
Sdílená	Na čipu	ne	ne	Čtení/Zápis	Blok	Blok
Globální	Mimo čip	2.x - ano	Čtení/Zápis	Čtení/Zápis	Aplikace	Aplikace
Konstant	Mimo čip	ano	Čtení/Zápis	Čtení	Aplikace	Aplikace

2.3 OpenGL

OpenGL je aplikační programovací rozhraní (API) pro renderování 2D a 3D grafiky, vyvinutý společností Silicon Graphic a vydaný v roce 1992. Před vytvořením OpenGL musel být psán unikátní kód pro grafickou aplikaci na každou platformu zvlášť. Za pomoci OpenGL je však možno vytvořit efekty vypadající ve výsledku téměř identicky na jakémkoliv operačním systému a grafické kartě, které podporuje OpenGL. Tato vlastnost je velmi důležitá. U softwaru vytvořeném pomocí OpenGL se nemusí řešit kompatibilita s různými softwarovými a hardwarovými platformami. Což ve výsledku značně ulehčuje práci [9, 10].



Ilustrace 9: OpenGL logo, http://www.opengl.org/

3 Implementace částicového systému

Pro vytvoření částicového efektu jsou použity technologie CUDA a OpenGL. Technologie CUDA provádí všechny náročné výpočty na GPU grafické karty. Mezi náročné výpočty se řadí pohyb jednotlivých částic uvnitř systému. Technologie OpenGL je použito jako prostředek pro vykreslení výsledku.

3.1 Praktická implementace

V této bakalářské práci je implementován částicový systém využívající výpočetní síly GPU grafické karty za pomoci technologie CUDA. Základní myšlenkou je vypočítání veškerých pohybů jednotlivých částic na GPU. Výsledné zpracované souřadnice jsou předány zpět do hlavní aplikace, kde jsou za pomoci OpenGL vykresleny. Pro samotnou implementaci bylo využito Microsoft Visual Studio 2008 s nainstalovanou 32 bitovou verzí CUDA 5.0. Jako vzor pro samotnou implementaci posloužil oficiální NVIDIA příklad, Simple OpenGL. Postup implementace by se dal rozdělit do tří hlavních kroků.

- Propojení technologií OpenGL a CUDA, inicializace dat
- Provedení výpočtů na CUDA zařízení
- Zobrazení výsledků pomocí OpenGL

3.1.1 Propojení CUDA a OpenGL

Pro implementaci částicového systému za pomoci CUDA a OpenGL je nejdříve nutné zajistit, aby spolu tyto technologie mohly spolupracovat. Pro rychlé předávání dat mezi zařízením a hostem slouží VBO, neboli Vertex object buffer. Prvním krokem k vytvoření tohoto spojení je vytvoření bufferu pro proměnnou VBO a specifikování jeho funkce. V tomto případě se jedná o *GL_ARRAY_BUFFER*, čímž je řečeno, že data v bufferu mají podobu pole. Buffer je následně zaregistrován a navázán na CUDA zařízení. Před samotným spuštěním CUDA výpočtů jsou zpracovány, tedy alokovány a naplněny, všechny relevantní proměnné a následně jsou překopírovány na proměnné umístěné na zařízení. Tímto způsobem jsou získána data pro první částice a náhodné hodnoty, které jsou využity pro výpočet pohybu částic.

```
for (int i = 0; i < particleNet_width * particleNet_height; i++)
  {
        particleDataHost [i].x = rand() % 500;
        particleDataHost [i].y = 0.0f;
        particleDataHost [i].z = 0.0f;
        particleDataHost [i].w = 0.0f;
  }
```

Jakmile jsou data a buffer uložena na zařízení, přichází na řadu samotné spuštění kernelu. Při

volání kernelu je specifikováno, kolik vláken a bloků kernel využívá. Po zavolání kernelu se všemi parametry, začne výpočet na GPU. Volání kernelu vypadá následovně. Data z něj budou přístupná přes ukazatel *dataPointerRegister*.

```
kernel<<<grid,block>>>(dataPointerRegister,particleDataDevice,particle
Net_width, particleNet_height,randomValuesDevice,MAXIMUM_AGE);
```

3.1.2 Výpočty na CUDA zařízení, kernel

Úkolem kernelu je vypočítání pozic částic, po provedení pohybu. Každá částice je dána pomocí id jednotlivých vláken. Tím je určeno, o jakou částici se jedná. Jako první na řadu přichází kontrola věku, kdy při překročení maximálního věku je pozice a věk částice nastaven na nulové hodnoty. Po kontrole věku přijde na řadu výpočet pohybu částice. Jednotlivé složky pohybu x, y, z jsou vypočítány funkcemi, které jsou následně volány v kernelu.

```
__device__ float calculateMovementX(float A, float B, float C, float
D, float X)
    {
      float movementX;
      movementX=((((0.0005*B)-(0.0005*A))+((A - C)/10000))/X);
      return movementX;
    }
```

Po vypočítání pohybu na jednotlivých osách, jsou tyto pohyby přičteny k aktuální pozici částic. Následuje kontrola, zdali částice nepřekročily požadované souřadnice. Pokud tak učiní, je jejich pozice vynulována. Je to dáno proto, aby se souřadnice částic nepohybovaly mimo vykreslovanou plochu. Na závěr je vytvořena nová float4 proměnná, kterou jsou vypočítaná data předána z kernelu na vykreslení.

```
positions[Loc] = make_float4(newX, newY, newZ, 1.0f);
```

Tím úkol kernelu končí.

3.1.3 OpenGL Zobrazení

Hlavním úkolem OpenGL je vytvoření okna a scény se všemi náležitostmi potřebnými pro zobrazení zpracovaných dat z kernelu. Zpracovaná data jsou přístupná přes VBO, která má nyní za použití *dataPointerRegister* namapovaná zpracovaná data v bufferu. Buffer obsahuje celkem čtyři hodnoty, tři z nich jsou vypočítané souřadnice a poslední hodnota je číslo jedna. Data jsou využita pro zobrazení nové pozice částice v každém cyklu, a také pro generování náhodné barvy. Výsledná scéna je v cyklu vykreslována, pokaždé s novými daty z bufferu.

4 Vizualizace částic a jejich interakcí

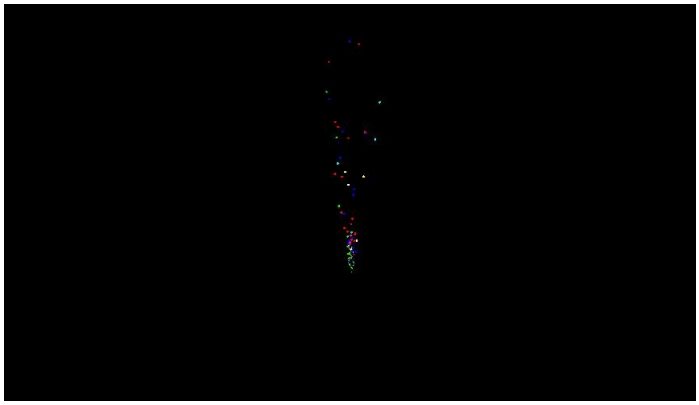
Vizualizace může být ve své konečné fázi součástí vytvořeného částicového systému. Je využívána k zobrazování různých objektů a efektů, tvořených z mnoha částic. V bakalářské práci je vytvořena scéna, která je naplněna pomocí částicového systému barevnými částicemi směrovanými v pohybu směrem vzhůru.

4.1 CUDA + OpenGL

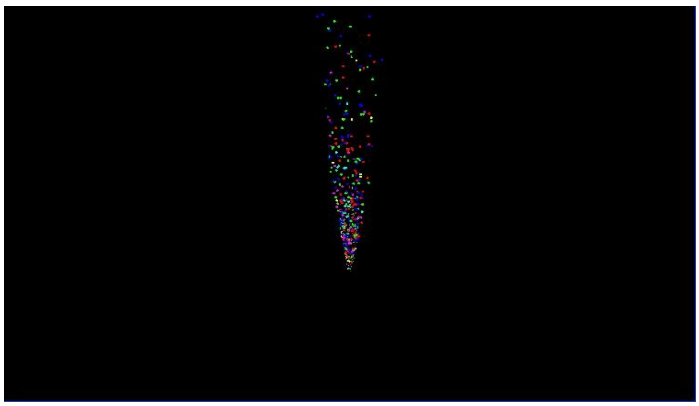
Zobrazování scény v implementovaném částicovém systému je provedeno pomocí technologie OpenGL. Základní princip propojení technologií OpenGL a CUDA je takový, že CUDA zvládá všechny náročnější výpočty pozic, různých fyzikálních jevů a podobně. Výsledná data jsou přes namapovaný buffer předány do OpenGL. Předaná data jsou ve formě čtyř čísel, kde tři čísla určí pozici částice v prostoru. OpenGL na základě dodaných dat zajistí vykreslení scény.

4.2 Vizualizace implementované části

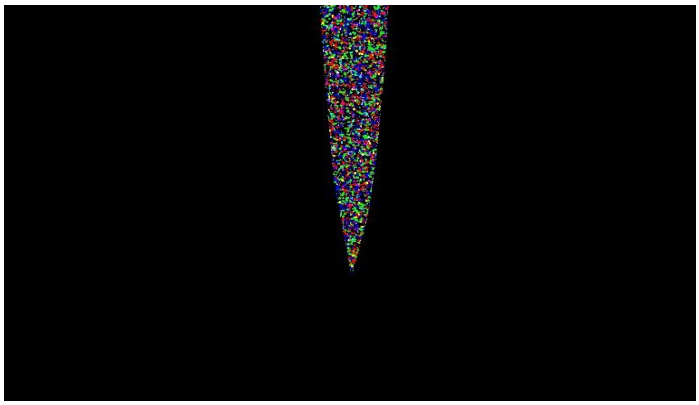
Jako součást této bakalářské práce byl vytvořen částicový systém, který simuluje pohyb částic. Částice se pohybují upravováním jejich <x, y, z> souřadnic, přičítáním nebo odečítáním čísla malé hodnoty, specifickým způsobem ovlivněné náhodnou hodnotou. Aktuálně je vizualizace implementována s principem rychlého stárnutí částic. Stárnutí částic je provedeno pomocí přičítání náhodné hodnoty k věku každé částice, tím je zajištěno to, že částice nežemřou ve stejný okamžik. Částice s nejčastěji přiřazovanou nízkou hodnotou mohou stárnout stárnou až 61 krát pomaleji, než ostatní elementy v systému. Rozsah pro stárnutí částic je tedy poměrně velký a jejich maximální věk není příliš vysoký. Velké množství částic umírá blízko startovní lokace. Výsledkem je vizuální dojem, že je na ploše v pohybu méně částic, než je v samotném systému obsaženo. Problém nejlépe vystihuje zdokumentovaná ilustrace 11, kde je v pohybu skutečně celkem 4096 částic. Samotné vizuální sledování daného jevu však skutečnému množství neodpovídá. Důvodem je zdůraznění toho, že tam opravdu pohyb částic probíhá. Bez uvedeného opatření by pro zahrnutý konkrétní efekt s nadměrným množstvím částic nebyl zřejmý žádný vizuální pohyb. Částice by splynuly do jednoho vizuálního celku.



Ilustrace 10: Zobrazení 256 částic



Ilustrace 11: Zobrazení 4096 částic



Ilustrace 12: Zobrazení 16 777 216 částic

5 Analýza a vyhodnocení výkonnosti implementace

V bakalářské práci je implementován výpočetně ne příliš složitý částicový systém. Nejsou počítány interakce mezi částicemi jako kolize a jiné matematicky náročnější operace. Změna pozice částic je počítána za pomoci náhodných čísel, která jsou předávána do kernelu od hosta.

5.1 Použitý hardware a software

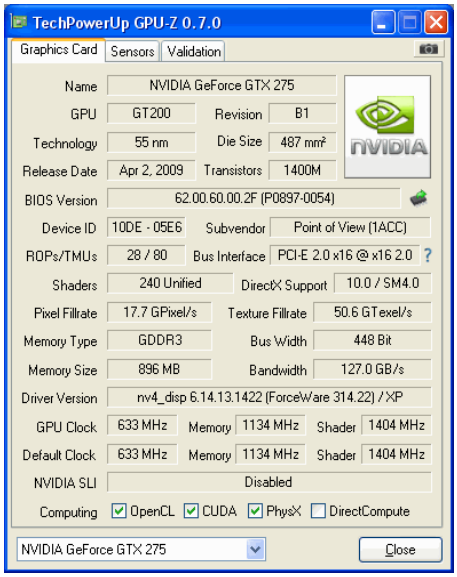
Následující měření probíhala na níže uvedené PC sestavě:

- Operační systém Windows XP 32bit SP3
- Procesor: Intel(R) Core(TM) i5 750, 2,67GHz
- Grafická karta: NVIDIA GeForce GTX 275
- Operační paměť: 3,49 GB RAM

Pro měření údajů týkajících se GPU byl použit freeware program GPU temp. Dokáže změřit vytížení jednotlivých částí grafické karty, a také pohlídat teploty, pokud by mělo dojít k přehřátí. Na získání FPS byl využit program Fraps. Měří FPS téměř v jakékoliv aplikaci.

5.1.1 Detaily grafické karty

Při měření výkonu implementace CUDA programu je hlavním parametrem ovlivňující výkon grafická karta. Pro tento test byla použita grafická karta NVIDIA GeForce GTX 275. Tato grafika nepatří mezi nejmodernější grafické adaptéry současnosti, i přesto skrývá poměrně velký výpočetní výkon a dala by se popsat jako velmi výkonná. Následují technické detaily zmíněné grafické karty. Pro jejich získání byl využit software TechPowerUp.



Ilustrace 13: Specifikace GPU

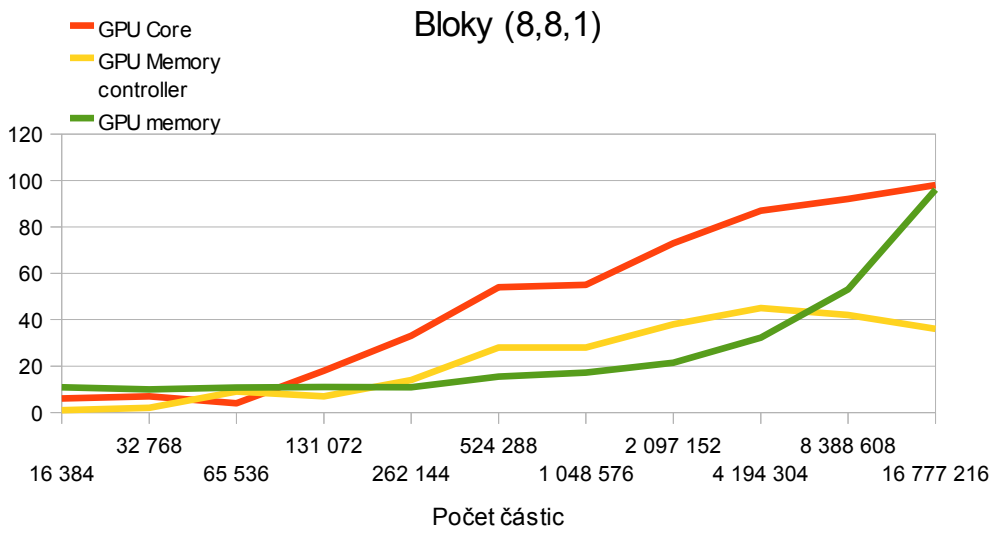
5.2 Měření hodnot

V praktickém měření výkonnosti byla měřena výkonnost CUDA a CPU implementace. Výkonnost implementace byla měřena pomocí zvyšování počtu aktivních částic uvnitř částicového systému. Částicový systém je vytvořen tím způsobem, že počet částic je zadáván pomocí dvou hodnot N,M. Výsledný počet částic je násobkem těchto hodnot. Hodnoty 128,128 znamenají 16384 částic uvnitř systému. Pro získání dostatečného počtu dat bylo provedeno celkem 11 měření na různých počtech částic. Měření začíná na 16384 částicích, v každém dalším měření je množství částic dvojnásobné, než v kroku předchozím. Výkon byl měřen na CUDA implementaci ve dvou různých nastaveních. Nastavení jednotlivých kroků měření spočívalo ve změně počtu aktivních vláken, které byly použity pro každý blok. Konkrétní nastavení bylo 64 vláken a v druhém případě 4 vlákna. Velikost mříže bloků se odvíjí od počtu částic. V další části měření byla použita CPU implementace stejného problému, tedy částicového systému obsahujícího určitý počet částic. Implementace využívá stejných výpočtů změny pohybu částic jako CUDA implementace. Pro stejné počty částic byla naměřena data a následné hodnoty srovnány s jednotlivými výsledky, jež dosahovala CUDA implementace.

5.2.1 Kernel blok (8,8,1)

První sada měření byla provedena na CUDA implementaci s nastaveným množstvím vláken pro jednotlivé bloky (8,8,1). Čísla v nastavení specifikují počet vláken v jednotlivých dimenzích. Výsledný počet vláken je vypočítán jako násobek těchto hodnot. Hodnoty (8,8,1) představují 8x8x1 vláken, tedy 64 vláken celkem. Velikost mříže bloků je při spuštění závislá na počtu aktuálně vkládaných částic do systému. Je-li počet částic definován například na hodnotu 128x128 a hodnota bloků na (8,8,1), tak výsledný počet mříží bude (128/8,128/8,1), tedy (16,16,1). Následují naměřené hodnoty pomocí programů Fraps a GPU temp.

Počet částic(NxM)	Počet částic	FPS	GPU Core[%]	GPU memory controller[%]	GPU memory[%]
128,128	16384	75	6	1	10,8
256,128	32768	75	7	2	10
256,256	65536	75	4	9	10,7
512,256	131072	75	18	7	11
512,512	262144	75	33	14	10,8
1024,512	524288	75	54	28	15,5
1024,1024	1048576	38	55	28	17,2
2048,1024	2097152	25	73	38	21,4
2048,2048	4194304	15	87	45	32,2
4096,2048	8388608	9	92	42	53,1
4096,4096	16777216	4	98	36	96

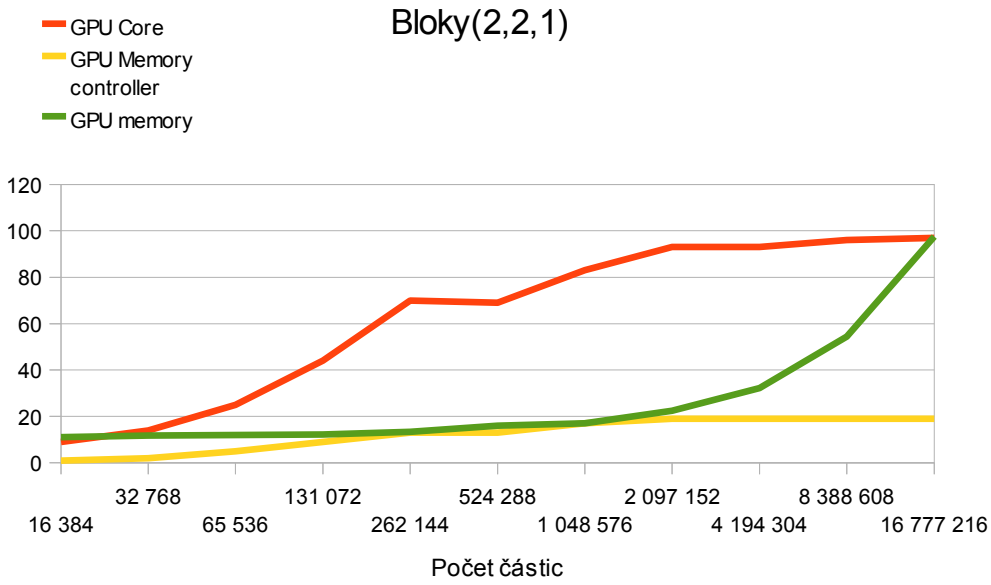


Graf 1: Vytížení pamětí a jádra grafické karty v poměru k počtu vykreslovaných částic za použití bloků velikosti (8,8,1).

5.2.2 Kernel blok (2,2,1)

Druhá sada měření byla provedena stejným způsobem jako sada první. Parametrem v testu byl počet aktivních částic uvnitř částicového systému. Rozdíl mezi měřeními je v počtu vláken přiřazených na jednotlivé bloky. V tomto případě bylo použito (2,2,1) vláken, tedy 4 vlákna na rozdíl od 64 vláken jako v první sadě měření. Následují naměřené hodnoty z druhé sady měření.

Počet částic(NxM)	Počet částic	FPS	GPU Core[%]	GPU memory controller[%]	GPU memory[%]
128,128	16384	75	9	1	11,1
256,128	32768	75	14	2	11,8
256,256	65536	75	25	5	12
512,256	131072	75	44	9	12,2
512,512	262144	68	70	13	13,3
1024,512	524288	38	69	13	16
1024,1024	1048576	19	83	17	17,1
2048,1024	2097152	11	93	19	22,5
2048,2048	4194304	5	93	19	32,2
4096,2048	8388608	3	96	19	54,4
4096,4096	16777216	1	97	19	97,5



Graf 2: Vytížení pamětí a jádra grafické karty v poměru k počtu vykreslovaných částic za použití bloků velikosti (8,8,1).

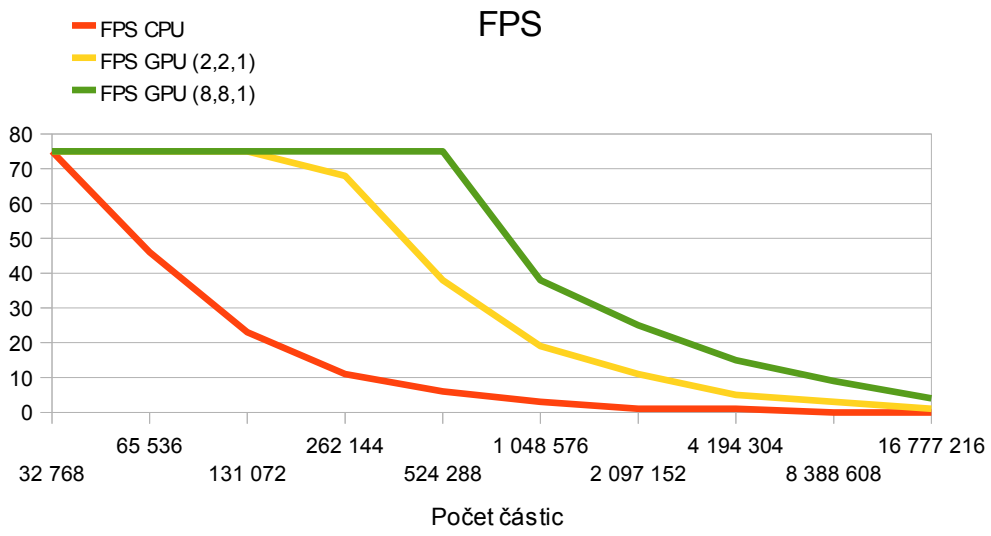
5.3 Porovnání

5.3.1 Kernel blok (8,8,1) a Kernel blok (2,2,1)

Z výsledných naměřených dat je možno vyčíst, že CUDA pracující s bloky kernelu o velikosti (8,8,1), dosahuje při řešení daného problému lepších výsledků, než za použití kernelu o velikosti (2,2,1). Rozdíl ve výkonu začíná být patrný od rozsahu částic 512x512, to je celkem 262144 jednotlivých částic. Při tomto množství částic se výkon začal rozcházet, což se projevilo vyšším využitím GPU jádra. Tento fakt přímo ovlivňuje FPS při vykreslování částic, které až do této chvíle bylo stálé na hodnotě 75 FPS. Hodnota 75 FPS je maximum, kterého při testu bylo možné dosáhnout. Hardware použitý při testu za nastaveného rozlišení nedovolí dosáhnutí vyšších hodnoty, protože maximální frekvence je pevně dána na 75Hz. Poslední testovanou hodnotou 4096x4096, neboli 16 777 216 částic, bylo téměř dosaženo výpočetních kapacit zařízení. Kernel s bloky (8,8,1) dokázal při těchto parametrech fungovat na 3 FPS, zatímco kernel (2,2,1) už dosahoval hodnot 1 FPS a menší.

5.3.2 CPU a GPU

Pro demonstraci výkonnosti zpracování pohybu částic částicového systému za pomoci CUDA s využitím výhod masivní paralelizace, byl implementován základní částicový systém pracující výhradně za pomoci CPU. Sledovaným parametrem při měření je FPS, za postupně zvyšujícího se počtu částic. Z výsledků měření je zřejmé, že CUDA technologie má díky paralelnímu zpracování dat při řešení tohoto konkrétního problému značnou výhodu.



Graf 3: FPS pro dvě GPU implementace s různým počtem použitých vláken a CPU implementaci.

6 Závěr

Cílem bakalářské práce bylo rozebrat problematiku týkající se částicových systémů jako celku, provést jeho praktickou implementaci pomocí technologie CUDA a následně vyhodnotit výkonnost provedené implementace. Po úspěšné implementaci částicového systému bylo zřejmé, že rychlost zpracování dat za použití masivní paralelizace má své výhody. Hromadné zpracování dat ve více vláknech je naprosto ideálním způsobem řešení problematiky částicových systémů.

Pro vyhodnocení výkonnosti provedené implementace pomocí technologie CUDA byl použit parametr FPS vykreslované scény, za použití ruzných počtů vláken. Výkonnost implementace CUDA byla porovnána s výkonem klasické CPU implementace z důvodu srovnání naměřených dat. Výkonnostní rozdíl začínal být patrný již kolem 30 000 částic v systému a se zvyšujícím se počtem částic se rozdíl ve výkonu zvyšoval. Následně bylo patrné, že CUDA implementace je několikanásobně výkonnější než implementace CPU.

Využívání GPU grafické karty technologií CUDA pro hromadné zpracování dat je dobrou ideou, která má své praktické využití. GPU programování je rozhodně oblastí s dobrým potenciálem do budoucna.

7 Literatura

[1] W. T. Reeves. 1983. Particle Systems—a Technique for Modeling a Class of Fuzzy Objects. ACM Trans. Graph. 2, 2 (Duben 1983), 91-108

[2] MARTIN, Allen. Particle Systems. Worcester Polytechnic Institute (WPI) [online]. 10.1.1997 [cit. 2013-05-03]. http://web.cs.wpi.edu/~matt/courses/cs563/talks/psys.html

[3] Particle system. Turkcebilgi English Section [online]. 15.2.2011 [cit. 2013-05-03]. http://english.turkcebilgi.com/Particle+system

[4] About :: GPGPU.org. GPGPU.org :: General-Purpose computation on Graphics Processing Units [online]. 3.3.2007 [cit. 2013-05-03]. http://gpgpu.org/about

[5] History of GPU Computing. Parallel Programming and Computing Platform | CUDA | NVIDIA [online]. 2013 [cit. 2013-05-03]. http://www.nvidia.com/object/cuda_home_new.html

[6] SWERTFEGER, Chris. Techware Labs - Articles - CUDA, What Is It and What Does It Mean To You?. TechwareLabs: Thinking Outside the Cube [online]. 25.9.2008 [cit. 2013-05-03]. http://www.techwarelabs.com/articles/editorials/CUDA/

[7] NVIDIA GT200: Inside a Parallel Processor. Real World Tech [online]. 8.9.2008 [cit. 2013-05-03]. http://www.realworldtech.com/gt200/2/

[8] VAN OOSTEN, Jeremiah. CUDA Memory Model |3D Game Engine Programming. 3D Game Engine Programming [online]. 25.11.2011 [cit. 2013-05-03]. http://3dgep.com/?p=2012

[9] OpenGL. TechTerms [online]. 2013 [cit. 2013-05-03]. http://www.techterms.com/definition/opengl

[10] What Is OpenGL. WiseGEEK: clear answers for common questions [online]. 3.4.2008 [cit. 2013-05-03]. http://www.wisegeek.com/what-is-opengl.htm

[11] David B. Kirk and Wen-mei W. Hwu. 2010. Programming Massively Parallel Processors: A Hands-On Approach (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

8 Příloha na CD

Obsah CD

- Bakalářská práce v pdf
  - bakalarska_prace.pdf
- Projekt Visual studio obsahující částicový systém.
  - kernel.cu
  - Castice.sln
  - Knihovny potřebné ke spuštění projektu
- Animace efektu
  - animace.avi
- Částicový systém CPU implementace
  - Particles.cpp
  - Knihovny potřebné ke spuštění projektu